Amir Montgomery, Insertion/Bubble Sort Assignment

This assignment was not too bad, I had a bit of trouble with the alternative bubble sort and deciding on how to go forward and backward. I ended up just making a direction bool that swapped after each go through, so it's not exactly odds = forward, evens = backwards, but it does alternate. I was confused on why results for two different lists came back so similar, but then I noticed it was always either randomized and ascending or randomized and descending – IE, my insertion sort algorithm is really good because the randomized (generated with Fisher-Yates algorithm, source code included) is really close to best case, whereas my bubble sort is terrible because the randomized is close to the worst case.

Alt Bubble Verification

```
amontg@LAPTOP-AMONTGO
Size: 8192, Avg. Comparisons: 3.3550336e+07
^Csignal: interrupt
amontg@LAPTOP-AMONTGOMERY:/mnt/c/Users/Amir Montgomery/Documents/S2023/Algorithms/Workspace$ go run main.go
Example List of Size 8: [1 2 3 4 5 6 7 8]
Size: 2, Avg. Comparisons: 1
Size: 4, Avg. Comparisons: 5
```

```go
    }
    if len(list) == 8 {
        fmt.Printf("Example List of Size 8: %v\n", list)
    }
    return comparisons
}

func DoAltBubbleSort() {
    tempSlice := []int{6, 2, 4, 7, 1, 3, 8, 5}
    AltBubbleSort(tempSlice, len(tempSlice))
```

Bubble Verification

```
amontg@LAPTOP-AMONTGO
Size: 1024, Avg. Comparisons: 523146
Size: 2048, Avg. Comparisons: 2.093782e+06
Size: 4096, Avg. Comparisons: 8.37683e+06
amontg@LAPTOP-AMONTGOMERY:/mnt/c/Users/Amir Montgomery/Documents/S2023/Algorithms/Workspace$ go run main.go
(Bubble) Example List of Size 8: [1 2 3 4 5 6 7 8]
Size: 2, Avg. Comparisons: 1
```

```go
    }
    if len(list) == 8 {
        fmt.Printf("(Bubble) Example List of Size 8: %v\n", list)
    }
    return comparisons
}

func DoBubbleSort() {
    tempSlice := []int{6, 2, 4, 7, 1, 3, 8, 5}
    BubbleSort(tempSlice, len(tempSlice))
```

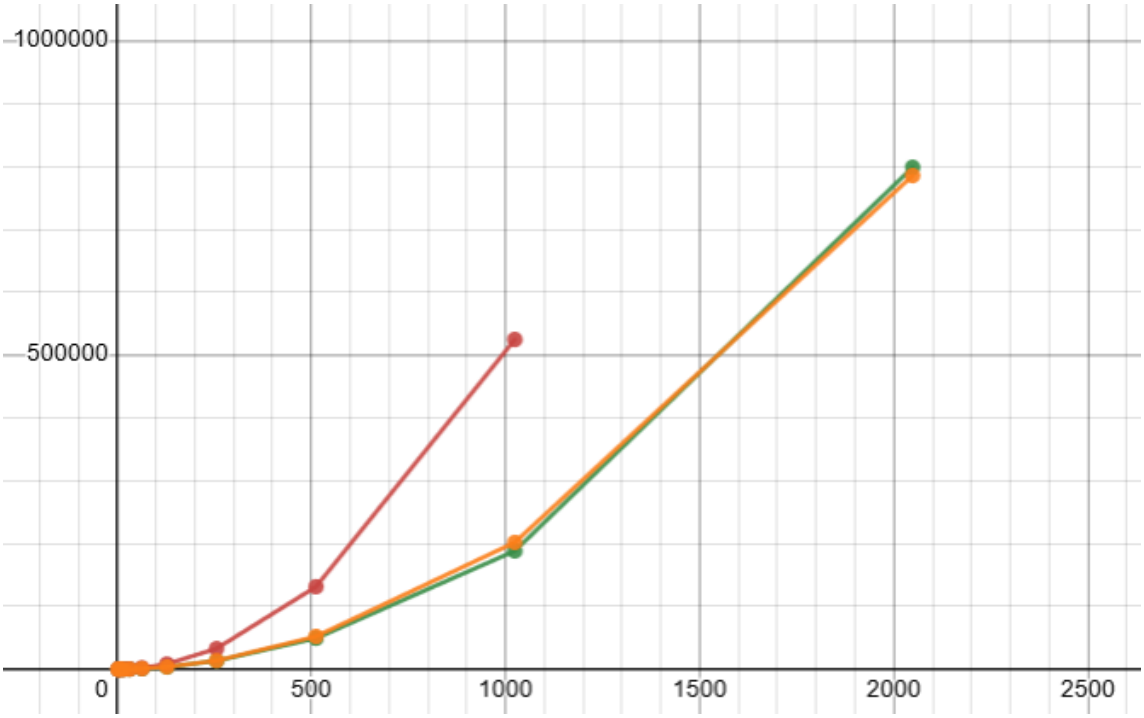
Insertion Verification

```
amontg@LAPTOP-AMONTGOMERY:/mnt/c/Users/Amir Montgomery/Documents/S2023/Algorithms/Workspace$ go run main.go
(Insertion) Example List of Size 8: [1 2 3 4 5 6 7 8]
Size: 2, Avg. Comparisons: 2
Size: 4, Avg. Comparisons: 6
Size: 8, Avg. Comparisons: 25
```

```go
    }
    if len(list) == 8 {
        fmt.Printf("(Insertion) Example List of Size 8: %v\n", list)
    }
    return comparisons
}

func DoInsertionSort() {
    // 2^(1-16)
    tempSlice := []int{6, 2, 4, 7, 1, 3, 8, 5}
    InsertionSort(tempSlice, len(tempSlice))
```

Insertion Sort



Ascending

x_1	y_1
2	3
4	7
8	27
16	63
32	248
64	826
128	3545
256	13319
512	49162
1024	188348
2048	798828

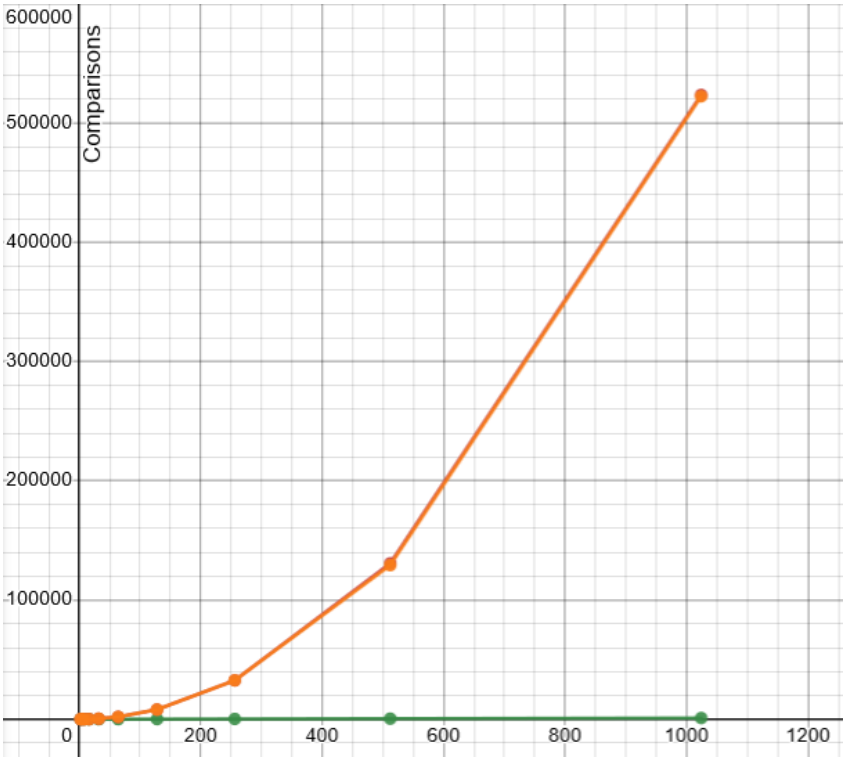
Descending

x_2	y_2
2	3
4	10
8	36
16	136
32	528
64	2080
128	8256
256	32896
512	131328
1024	524800

Randomized

x_3	y_3
2	3
4	7
8	25
16	71
32	238
64	1026
128	3662
256	13632
512	52234
1024	201889
2048	785707

Bubble Sort (Descending plot is right under Randomized)



Ascending

x_1	y_1
2	1
4	3
8	7
16	15
32	31
64	63
128	127
256	255
512	511
1024	1023

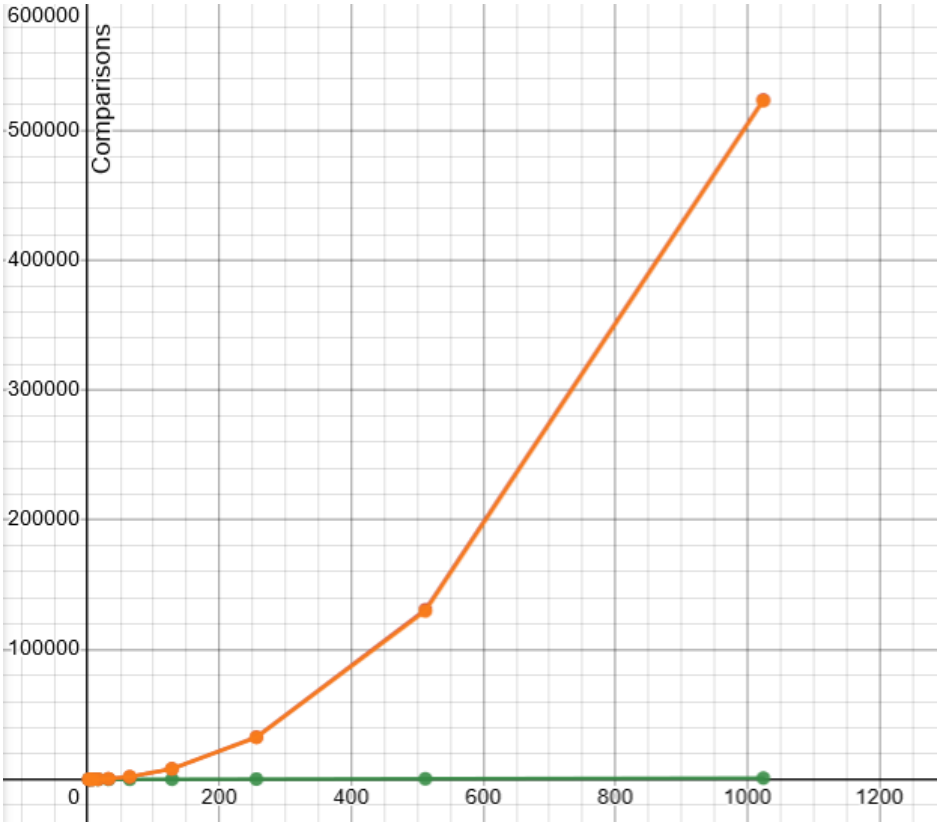
Descending

x_2	y_2
2	1
4	6
8	28
16	120
32	496
64	2016
128	8128
256	32640
512	130816
1024	523776

Randomized

x_3	y_3
2	1
4	6
8	28
16	84
32	460
64	1980
128	8073
256	32585
512	129220
1024	522568

Alt. Bubble Sort (Same deal as regular bubble sort)



Ascending		Descending		Randomized	
x_1	y_1	x_2	y_2	x_3	y_3
2	1	2	1	2	1
4	3	4	6	4	5
8	7	8	28	8	25
16	15	16	120	16	117
32	31	32	496	32	495
64	63	64	2016	64	2013
128	127	128	8128	128	8113
256	255	256	32640	256	32367
512	511	512	130816	512	129734
1024	1023	1024	523776	1024	523146